METHOD AND SYSTEM FOR GENERATING A TRANSFORM

Related Applications

This patent application claims priority on provisional patent application 60/240,578 filed on October 13, 2000, entitled "Optimized Coding Methods for Icon Generation and Manipulation in DPP" and assigned to the same assignee as the present application.

Field of the Invention

The present invention relates generally to the field of associative memories, associative processing and polynomial code generation and more particularly to a method and system for generating a transform.

Background of the Invention

Associative memories and associative processing as described in US patent Nos. 5,942,002; 5,742,611; 6,157,617: 6,617,400; and US patent application Nos. 09/419,217; 09/672,754; 09/768,102; 09/767,797; 09/768,101; 09/767,493 all assigned to the same assignee as the present application, require transform generators (polynomial

code generators). As explained in the above referenced patents (patent applications) a transform (polynomial code) is generated for use in an associative memory. The polynomial generator is generally applied to a key to determine an address (transform, CRC, remainder, code) in a memory where information relating to the key is stored. In the above applications the transform (CRC, remainder) is broken into an address and a confirmer. The confirmer is used to resolve collisions between two distinct items that might otherwise have the same address. In addition to associative memory applications, associative processing applications are described in the above referenced patents (patent applications). Both of these applications require the fast and efficient generation of transforms (polynomial codes). Two general methods of polynomial code generators are known. One is using linear feedback shift registers to perform the division modulo two on a bit by bit basis. There have been some efficiencies realized in the general process, but the basic idea is the same. The other method involves a table lookup process. This method has generally been limited to a byte by byte process. This is because the lookup table for a byte range of number is 256 (2^8) entries. If the table were to be expanded for two bytes the number of entries would be 65,536 entries. Obviously a table this large is slow are requires a lot of memory. Despite this there is a need for a two bytes by two bytes process when oriental symbols encoded, such as used in Japan and China. All the potential symbols in these languages cannot be encoded by a single byte. Another use for more than one byte at a time is for computers that now have 64 bit words.

Thus there exists a need for an improved method and system for generating a transform (polynomial code, CRC) that is faster and more efficient than previous systems and methods.

Brief Description of the Drawings

FIG. 1 is a block diagram of a system for generating a transform in accordance with one embodiment of the invention;

FIG. 2 is a block diagram of a system for generating a transform in accordance with one embodiment of the invention;

FIGs. 3 & 4 are a flow chart of the steps used in generating a transform in accordance with one embodiment of the invention; and

FIGs. 5-8 are examples of lookup tables that might be used in the system of FIG. 1 in accordance with one embodiment of the invention.

Detailed Description of the Drawings

A system for generating a transform includes a first transform lookup table and second transform lookup table. A transform exclusive OR array is connected to an output of the first transform lookup table and an output of the second transform lookup table. The system allows transforms (polynomial codes, CRCs) to be generated using two or more tables. This means two bytes can be processed at a time without having to use a lookup table with 65,536 entries. Instead, two tables with 256 entries can be used. This is a much more efficient and faster way to calculate a transform.

FIG. 1 is a block diagram of a system 10 for generating a transform in accordance with one embodiment of the invention. The system has a first lookup table 12 connected to an exclusive OR (first exclusive OR) 14. The output 16 of the exclusive OR is connected to an address register (pointer) of the first lookup table 12. The inputs to the exclusive OR 14 are the first eight bits (byte) 17 of the input data 18 and the least significant byte 20 from the output register. The output 24 of the first lookup table 12 is connected to a transform exclusive OR 26. Note that the term "transform" is used to distinguish this exclusive OR 26 from the other exclusive ORs such as exclusive OR 14.

A second lookup table 28 is connected a second exclusive OR 30. The second exclusive OR 30 has a first input 34 connected to the second byte of input data 18 and a second input 36 connected the second byte of the output register 22. The output 32 of the lookup

table 28 is connected to the transform exclusive OR 26. A third lookup table 38 is connected a third exclusive OR 40. The third exclusive OR 40 has a first input 42 connected to the third byte of input data 18 and a second input 44 connected the third byte of the output register 22. The output 46 of the lookup table 38 is connected to the transform exclusive OR 26. A fourth lookup table 46 is connected a fourth exclusive OR 48. The fourth exclusive OR 48 has a first input 50 connected to the fourth byte of input data 18 and a second input 52 connected the fourth byte of the output register 22. The output 54 of the lookup table 46 is connected to the transform exclusive OR 26. The transform exclusive OR 26 is a five term exclusive OR array. The fifth term is the output or code register 22 shifted to the right by 32 bits. In other words the four least significant bytes are discarded and the rest of the transform is shifted four significant bytes less. This shift is equal to the slice of input bits processed.

Using the system 10 up to thirty-two bits of data can be processed into a transform per cycle. Note the process can be iterative, so that any length of data string can be processed four bytes at a time. In addition, it will be apparent to those skilled in the art that the number of tables can be expanded from one table to as many tables as desired. FIGs. 5-8 show an example of four lookup tables generated using the divisor polynomial whose coefficients are:

1E543279765927881 hex

This results in a 64 bit transform (icon, polynomial code). Other transform lengths may also be used. Note one limitation on expanding the system is that the input data length (slice of data) may be no longer than the transform length. Table one 12 is calculated by shifting the first byte range (00 - FF hex) by 64 bits (multiplying by 2^{64}) and determining a transform for every number in the range. Table two 28 is calculated by shifting the first byte range by 64 bits plus 8 bits (multiplying by 2^{72}). Table three is calculated by shifting the first byte range by 64 bits plus 16 bits (multiplying by 2^{80}). Table four is calculated by shifting the first byte range by 64 bits plus 24 bits (multiplying by 2^{88}). The invention is not limited to any particular polynomial divisor, however the divisor's that are irreducible are generally preferable.

As will be apparent to those skilled in the art, the tables could be setup for 4 bit slices or any number of bits. However from a practical stand point it makes sense to have tables based on some multiple of a byte (1/4, 1/2, 1, 2, 4).

FIG. 2 is a block diagram of a system 70 for generating a transform in accordance with one embodiment of the invention. This system is optimized to determine a transform (code, CRC) for a sliding window. For instance if a string of data bytes (EDCBA) is received, the system would determine the transform for AB noted as X(AB), and X(BC), X(CD), X(DE). This is particularly helpful when a user is attempting to perform a search or scan of the incoming data. This application is discussed in more detail in US Patent No. 6,167,400, entitled "Method of Performing a Sliding Window Search".

The system 70 has a first lookup table (append character lookup table) 72 connected to an exclusive OR (data exclusive OR) 74. One input to the data exclusive OR 74 is a new portion 76 of a data string 78 and the second input is connected to a portion 79 (least significant portion) of the present transform (output register) 80. An output 82 of the first lookup table 72 is connected to an exclusive OR array 84. A second lookup table (remove character lookup table) 86 has an input connected to a shift register 88. This effectively connects the second lookup table 86 to a discarded portion of the data string 78. An output 90 of the second lookup table 86 is connected to the exclusive OR array 84. The output register 80 receives the result of the three term exclusive OR array 84 and the output of the output register 80 is shifted right (least significant) by eight bits and input into the exclusive OR array. Note that the amount of shift is equal to the number bits in the new portion 74 and the new portion of data is equal in bits to the discarded portion.

In order to understand how the system 70 works some basic understanding of transform algebra (icon algebra, linear algebra) is required. A more formal treatment of the underlying math can be found in the Appendix of the provisional patent application 60/240,578 filed on October 13, 2000, entitled "Optimized Coding Methods for Icon Generation and Manipulation in DPP" and assigned to the same assignee as the present application. The present application claims priority based on this provisional application. Assume we have a data stream (EDCBA), wherein each letter represents a byte of data. Further assume we want to find two byte

combinations. The system will then be designed to produces a sliding transform of two bytes: X(AB), X(BC), X(DC) and X(ED). Assume that the output register contains the transform X(AB). Then the least significant eight bits 79 of the transform X(AB) are exclusive ORed 74 with the input data "C". This forms a pointer into the first table 72. If the transform is thirty two bits, then we will represent the output from the append table as $A_1A_2A_3A_4$ and the transform of X(AB) as $T_1T_2T_3T_4$. The transform X(AB) is shifted right by eight bits before being placed in the three term exclusive OR array 84. So this result in 0 $T_1T_2T_3$ being exclusive ORed with $A_1A_2A_3A_4$, where "0" represents a byte of zeros. The result of this exclusive OR operation is the transform X(ABC). This is a well know process used in generating CRCs using a table lookup. The next step is to exclusive OR this transform X(ABC) with the output 90 of the remove character lookup table. In this example the shift register 88 would be two bytes (the same length as the underlying data being transformed). Thus the input to lookup table two 86 is the byte "A". The second lookup table will produce the transform for A00 or X(A00), where the "0" represents a byte of zeros. Next we exclusive OR X(A00) with X(ABC). From transform algebra we know this results in the transform X(BC). For a detailed explanation of the math see the above referenced provisional patent appendix. An informal proof of how this works starts by noting the A XOR A is 0 where "A" and "0" represent bytes of data. Thus X(ABC) is the same thing as X(AX^{16} XOR B X^{16} XOR C), where X^{16} means multiply by 2^{16}. Then X(A00) XOR X(ABC) is the same thing as X(A00 XOR ABC) or

X(0BC) which is X(BC). The reason we can move the arguments inside the transform is because the transform is nothing more than a number of exclusive OR operations.

It is possible to combine the concepts of FIG. 1 with the concepts of FIG. 2 to have two or more append tables (plurality of tables) and have two of more remove tables (plurality of tables). As will be apparent to those skilled in the art, this would requires a larger exclusive OR array 84.

FIGs. 3 & 4 are a flow chart of the steps used in generating a transform in accordance with one embodiment of the invention. The process starts, step 100, by selecting a generator polynomial at step 102. A plurality of lookup tables are generated using the generator polynomial at step 104. A data string is received at step 106. The data string is divided into a plurality of data portions, one for each of the plurality of tables at step 108. A lookup is performed in each of the plurality of lookup tables to find a plurality of partial transforms based on the plurality of data portions at step 110. At step 112, a transform is created based on the plurality of partial transforms which ends the process at step 114. When the data string is longer than the input to the plurality of tables, a second plurality of data portions of the data strings. Each of the second plurality of data portions is exclusive ORed with a portion of the transform to form a plurality of pointers is selected. A lookup is performed in each of the plurality of lookup tables using the plurality of pointers to find a second plurality of partial transforms. The transform is multiplied by a factor to form a moved transform. The moved transform is

exclusive ORed with the second plurality of partial transforms to form a new transform. Note the factor that the transform is multiplied by is 2^{-x} where x is equal to the number of input bits per iteration. The resulting number is rounded off to a whole number.

In one embodiment the step of generating a plurality of lookup tables includes the step of selecting a range of numbers. Moving the range of numbers by a number of bits equal to the number of bits in the transform to form a shifted range of numbers. Next the shifted range of numbers are divided modulo n (where n is commonly 2) by the generator polynomial to form a plurality of entries for one of the plurality of lookup tables. In one embodiment the range of numbers is a byte of numbers.

Thus there has been described a method and system for creating a transform that is faster and more efficient than previous methods and systems. This increases the efficiency and usefulness of associative memories and associative processing.

The methods described herein can be implemented as computer-readable instructions stored on a computer-readable storage medium that when executed by a computer will perform the methods described herein.

While the invention has been described in conjunction with specific embodiments thereof, it is evident that many alterations, modifications, and variations will be apparent to those skilled in the art in light of the foregoing description. Accordingly, it is intended to embrace all such alterations, modifications, and variations in the appended claims.